Cole Bardin
CS361
10/3/24

HW2

Question 1:
a)
There are 2 possible outputs:
1 - “PQ”
At the start, both threads attempt to acquire the lock. If P acquires the lock first, it will print then release the lock, letting Q move on. Thus the output will be “PQ”
2 - “QP”
At the start, both threads attempt to acquire the lock. If Q acquires the lock first, it will print then release the lock, letting P move on. Thus the output will be “QP”

b)
In this case, if P gets the lock first, it will print out “P”. But since it never releases the lock, it Q will be deadlocked and never return. But if Q acquires the lock first, it will print and then release the lock. Then P will acquire the lock and print. In this case, both threads will return and the results will be “QP”

Question 2:
a)
The minimum number of * can be 0. This would happen if P acquires the lock first, sets B to True then releases the lock and returns. Then Q will be allowed to acquire the lock however B is True so the while loop does not execute. Then Q will just return and no * will be printed.

The other option would be that Q gets the lock first and then since B is false and P is locked awaiting for Q to release the lock, Q will infinitely print out *. Q will be livelocked and P will be deadlocked.

Question 3:

Algorithm Q3: Signaling	
Setup: Integer n = 0, Lock lk, lk.lock()	
Thread P	Thread Q
P1: for i from 1 to 20 do: P2: temp = n	Q1: lk.lock() Q2: print(n)

P3: n = temp + 1 P4: lk.unlock()	

My solution involves adding a lock that is initialized and locked in the setup. Having Q attempt to acquire the lock but now P will make it so that P always executes first. Then when P is done, it releases the lock and allows Q to print out the last value of n.